# Recurrent Neural Network (RNN) Algorithm Outline

## Overview

This document outlines the algorithmic steps and mathematical equations involved in the forward and backward propagation of a Recurrent Neural Network (RNN), including weight updates and the overall training loop.

## 1. Initialization

1. **Initialize Parameters:** - Weight matrices:

$$W_x \in \mathbb{R}^{\text{hidden\_size} \times \text{input\_size}}, \quad W_h \in \mathbb{R}^{\text{hidden\_size} \times \text{hidden\_size}}, \quad W_y \in \mathbb{R}^{\text{output\_size} \times \text{hidden\_size}}$$

- Bias vectors:

$$b_h \in \mathbb{R}^{\text{hidden\_size}}, \quad b_y \in \mathbb{R}^{\text{output\_size}}$$

2. **Initialize Hidden State:**

$$h_0 = \mathbf{0}$$

where $h_0$ is a vector of zeros of size hidden_size.

—

## 2. Forward Propagation

For a sequence of inputs $X = \{x_1, x_2, \ldots, x_T\}$:

1. **Pre-activation for the Hidden State:**

$$z_t = W_x x_t + W_h h_{t-1} + b_h$$

where:

- $x_t \in \mathbb{R}^{\text{input\_size}}$: Input at time step $t$.
- $h_{t-1} \in \mathbb{R}^{\text{hidden\_size}}$: Hidden state from the previous time step.

2. **Hidden State Update:**

$$h_t = f(z_t)$$

where $f$ is an activation function such as ReLU, Sigmoid, or Tanh.

3. **Output Computation:**

$$y_t = W_y h_t + b_y$$

4. **Output Activation:**

$$\hat{y}_t = g(y_t)$$

where $g$ is an output activation function such as Softmax (for classification) or Identity (for regression).

—

## 3. Backward Propagation

Backward propagation computes the gradients of the loss function $L$ with respect to the network parameters.

1. **Gradient of the Loss w.r.t. the Output:**

$$\frac{\partial L}{\partial \hat{y}_t} = \hat{y}_t - y_t^{\text{true}}$$

2. **Gradients w.r.t. Output Weights and Biases:**

$$\frac{\partial L}{\partial W_y} = \sum_{t=1}^{T} \frac{\partial L}{\partial \hat{y}_t} \cdot h_t^\top$$

$$\frac{\partial L}{\partial b_y} = \sum_{t=1}^{T} \frac{\partial L}{\partial \hat{y}_t}$$

3. **Gradients w.r.t. Hidden States:**

- For $t = T$ (last time step):

$$\frac{\partial L}{\partial h_T} = W_y^\top \cdot \frac{\partial L}{\partial \hat{y}_T}$$

- For $t = T-1, \ldots, 1$:

$$\frac{\partial L}{\partial h_t} = W_y^\top \cdot \frac{\partial L}{\partial \hat{y}_t} + W_h^\top \cdot \frac{\partial L}{\partial h_{t+1}}$$

4. **Gradients w.r.t. Weights and Biases:** - Input-to-Hidden Weights:

$$\frac{\partial L}{\partial W_x} = \sum_{t=1}^{T} \frac{\partial L}{\partial z_t} \cdot x_t^\top$$

- Hidden-to-Hidden Weights:

$$\frac{\partial L}{\partial W_h} = \sum_{t=1}^{T} \frac{\partial L}{\partial z_t} \cdot h_{t-1}^\top$$

- Hidden Bias:

$$\frac{\partial L}{\partial b_h} = \sum_{t=1}^{T} \frac{\partial L}{\partial z_t}$$

where:

$$\frac{\partial L}{\partial z_t} = f'(z_t) \odot \frac{\partial L}{\partial h_t}$$

—

## 4. Weight and Bias Updates

Using the computed gradients, update all weights and biases using Gradient Descent:

$$\theta \leftarrow \theta - \eta \cdot \frac{\partial L}{\partial \theta}$$

where $\eta$ is the learning rate and $\theta$ represents any parameter $(W_x, W_h, W_y, b_h, b_y)$.

—

## 5. Training Loop

The overall training loop for the RNN is as follows:

1. **Initialization:**
    - Initialize weights, biases, and hidden states.
2. **Forward Propagation:**
    - Compute $z_t$, $h_t$, $y_t$, and $\hat{y}_t$ for all time steps.
3. **Backward Propagation:**
    - Compute gradients $\frac{\partial L}{\partial \theta}$ for all parameters.
4. **Parameter Update:**
    - Update $W_x$, $W_h$, $W_y$, $b_h$, and $b_y$.
5. **Repeat:**
    - Loop through multiple epochs or batches until convergence.

—

## Conclusion

This document outlines the key mathematical equations and algorithmic steps for forward and backward propagation in an RNN. These computations form the foundation for training the network on sequential data.